PID MOTOR CONTROL WITH THE Z8PE003

TABLE OF CONTENTS

Acknowledgements

Project Lead Engineer
Ken Johnston

Application and Support Engineers
Steve Narum, Greg Bergeson, Scott Bowden

System and Code Development
Ken Johnston

ABSTRACT

The Z8PE003 Microcontroller (MCU) is a One Time Programmable (OTP) member of ZiLOG's single-chip Z8Plus™ MCU family. It allows easy software development and prototyping, and because of its high clock rate and fixed instruction execution time, the Z8PE003 offers excellent computation speed. Being a low-cost solution makes it an excellent choice for basic control applications.

This application note describes the use of a Z8PE003 as a digital DC servo-motor controller using the common Proportional-Integral-Differential (PID) filter and demonstrates the effects of changing the PID constants and the immediate effects on the control loop.

This application note describes theory of operation, hardware, firmware, and software.

DISCUSSION

Theory of Operation

Digital Closed-Loop Theory

A closed-loop control system depends on the output of the process to adjust the signal controlling the closed loop. A feedback device measures the process output. The process output is compared to the user command and an output from the plant is issued. Figure 1 illustrates the functional block diagram of a closed-loop system. The plant controls the process and determines any error between what was commanded and what is actually happening. The process may be any controllable quantity such as voltage, current, speed, torque, temperature, or, as in this application note, position.

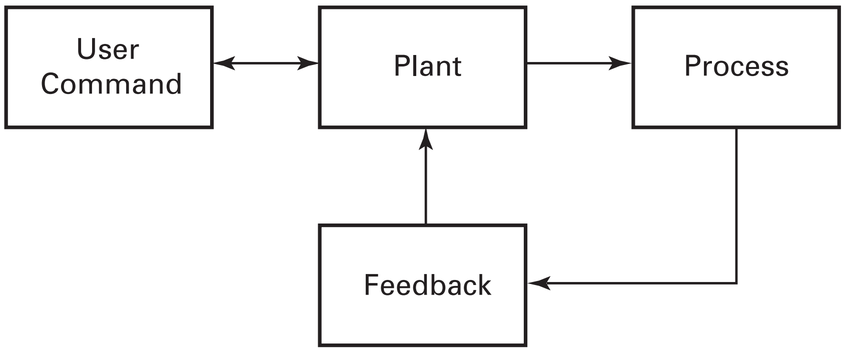

Figure 1. Block Diagram of a Closed-Loop System

In closed-loop mode, user input determines the *position command*. The feedback giving the *error* (see Figure 2) determines the actual position. The *error value*, U(k), is then applied to the digital PID filter as shown in Figure 3.

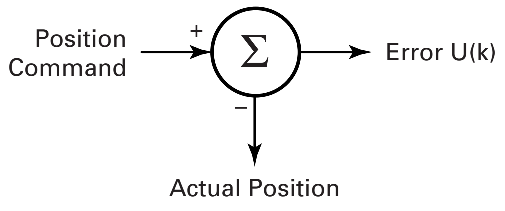

Figure 2. Error Signal

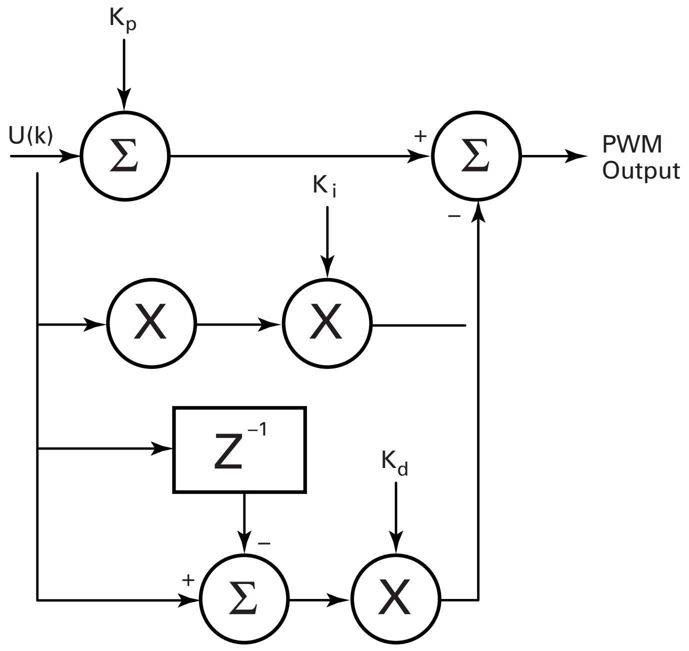

Figure 3. PID Filter

Figure 3 is the Z-transform of a continuous PID filter design. The designer determines the control constants K_p, K_i, and K_d through system stability analysis. Adjusting the control constants until optimum performance is obtained is called tuning the PID filter.

The stability of the feedback system is critical because it directly affects the process output. Performance that exhibits excessive overshoot, or underdamped, characteristics is unsatisfactory in many applications. The goal of tuning the PID filter is to obtain a critically damped system.

A critically damped system provides optimum tracking, settling time, and overshoot. There are many mathematical stability analysis techniques available to a well-modeled system, including the Laplace method, frequency-response, Nyquist criterion, Bode representation, root-locus method, and State-space method. Discussions of these techniques are beyond the scope of this application note.

Visual methods of tuning the PID filter also are widely used. The designer's first step is to make coarse adjustments of the constants to obtain reasonably good response characteristics. The second step fine tunes the constants with a measurement device, such as an oscilloscope.

Open-Loop Theory

An open-loop control system is one in which the control signal of the process is independent of the process output. The control accuracy is determined by the calibration of the plant. Figure 4 contains the functional block diagram of the open-loop system. This application note describes two different open-loop techniques using manual input switches. The user determines the position or velocity of the motor by pushing the manual input buttons.

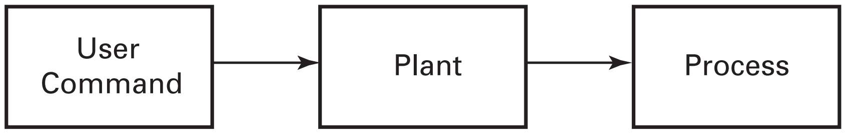

Figure 4. Block Diagram of an Open-Loop System

HARDWARE IMPLEMENTATION

Figure 5 contains the hardware functional block diagram. The sections following Figure 5 describe each block and its function. Appendix A contains the schematic diagram related to the block diagram.

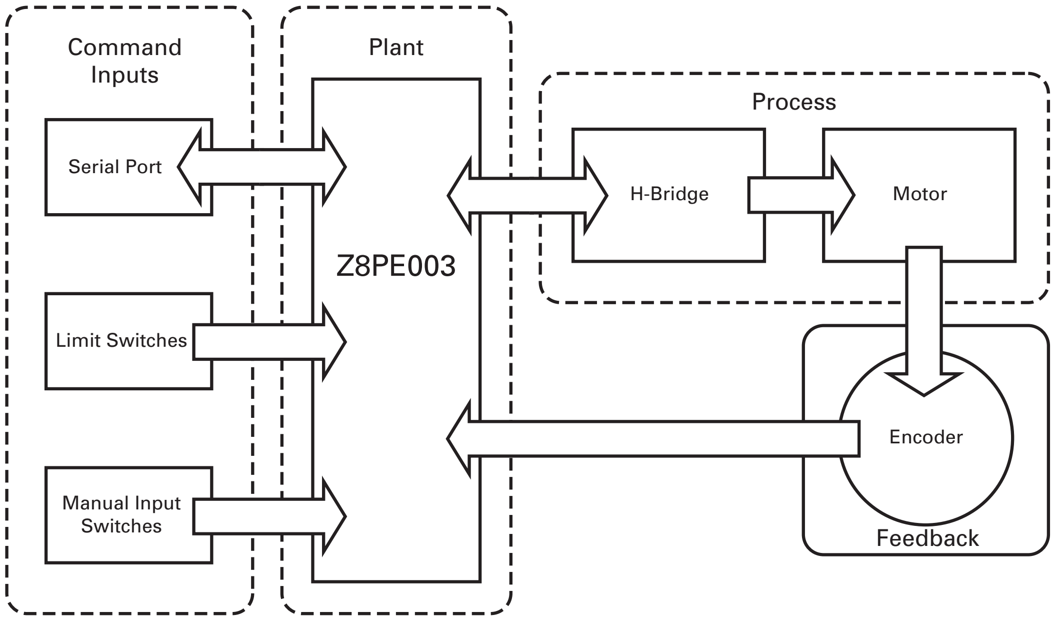

Figure 5. Hardware Functional Block Diagram

Plant

The Z8PE003 is the plant of the control loop. The Z8PE003 receives the command inputs, monitors the sensors, drives the H-Bridge, and updates the GUI on the current status. The Z8PE003's pin-out, direction and function are described in Table 1. Port pins PA0 and PA1 are used for the input from the manual switches. Serial communication is from the connections on PA2 and PB4. Port pins PA3, PA6, PA7, PB2, and PB1 control and monitor the H-Bridge. PB0 and PB3 are the encoder input port pins. Limit switches are tied to input pins PA4 and PA5.

Table 1. Z8PE003 Pin Functionality

Port	Direction	Description
PA0	IN	Manual input switch
PA1	IN	Manual input switch
PA2	OUT	Serial output
PA3	OUT	H-Bridge direction bit
PA4	IN	Limit switch input
PA5	IN	Limit switch input
PA6	IN	H-Bridge Thermal Flag
PA7	OUT	H-Bridge Brake Output
PB0	IN	Encoder Input
PB1	OUT	PWM Output
PB2	IN	Current Limit Input
PB3	IN	Encoder Input
PB4	IN	Serial Input

Command Inputs

Command inputs from the user are received through the serial connection. These commands are described in the firmware and hardware implementation sections of this application note. The other command inputs used by the Z8PE003 are manual input switches. The limit switches also provide a command input to the Z8PE003 to stop the motor. These limit switches are only used in open-loop mode. Implementation of the limit switches is described in the Firmware section. The plant interfaces with current limit and thermal overload flags that are described in the Process section.

Serial Communication

The RS-232 port is driven by a dedicated serial port transceiver. The serial interface uses a two-wire interface consisting of a receive line and a transmit data line (see schematic in Appendix A). Note that RD is the received data line and TD is the transmitted data line seen from the PC serial port perspective. The firmware and software implementation sections of this document describe implementation details.

Manual Input Switches

Manual input switches are used for open-loop control of position or velocity. These switches are momentary push-button single-pull/single-throw (SPST). When the user activates a switch, it pulls the input pin PA0 or PA1 Low. When not pushed, the input pins are pulled High by a resistor. LEDs D3 and D4 indicate that the user activated the switches.

Limit Sensors

The limit sensors consist of a slotted detector with an optical IR light-emitting diode (LED) and a phototransistor in a plastic housing. The signal is broken when an object is passed through the gap in the housing. Breaking this transmit signal turns off the base current to the phototransistor, turning off the phototransistor. The output signal is then pulled to a logic High level on the Z8PE003 (see Figure 6).

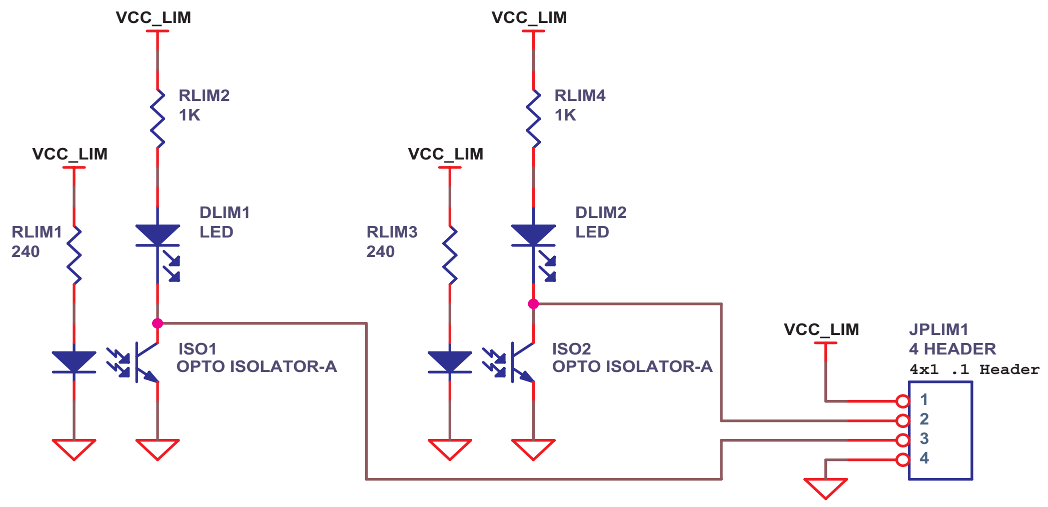

Figure 6. Limit Switch Functional Schematic

The LED turns on the phototransistor when there is no object in the optical path, pulling the output signal to a Low. A resistor in series with the LED limits the current to 18 mA. LEDs DLIM1 and DLIM2 illustrate when the phototransistor is turned on.

Feedback

The encoder interfaces with the plant as a feedback device. The encoder indicates the current position of the motor.

Encoder

The encoder chosen for this application is a rotary optical encoder that returns a 2-bit gray code. See Table 2.

Channel A	Channel B
0	0
0	1
1	1
1	0

Table 2. Gray Code Output

The encoder produces 128 pulses per revolution per channel. The two channels combine to produce 512 counts per revolution. See Figure 7.

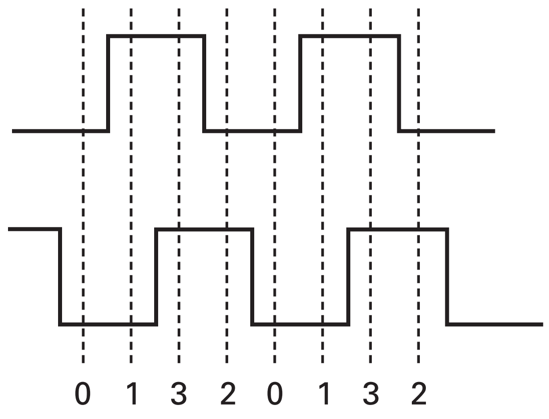

Figure 7. Encoder Output

The 512 counts allow a 0.7 degree per count value, which corresponds to a measurement accuracy of ±0.005 inches, as follows:

```
Circumference = p x Diameter of Gear = p x 0.875 inches = 2.749 inches
Encoder Control Value = 0.7 x ∏ pulse x (2.749 inches ∏ 360°) = ± 0.005
```

The slack in the belt accounts for a much greater error than the encoder. Experiments for the belt error determined an estimated value of ±10 counts. LEDs D5 and D6 are connected to the encoder output signal to indicate rotation of the encoder.

Process

The process described in this application note is the movement to a required position using a motor. The plant controls the H-Bridge with pulse width modulation (PWM). The H-Bridge output controls the motor.

H-Bridge

The H-Bridge for this application is the LMD18200 from National Semiconductor. The H-Bridge provides up to 3A of continuous current with a very low Rds(on) of 3Ws per switch. The Z8PE003 controls the direction, brake, and PWM inputs.

H-Bridge configurations use four Bipolar Junction Transistor (BJT) or Metal Oxide Silicon Field Effect Transistor (MOSFET) devices configured in an *H* pattern as shown in Figure 8. The motor is in the center of the *H*. Current drives the motor by flowing through Q1 and Q4 for one direction or Q2 and Q3 for the opposite direction. The built-in direction bit of the H-Bridge controls the current flow through the different MOSFETs.

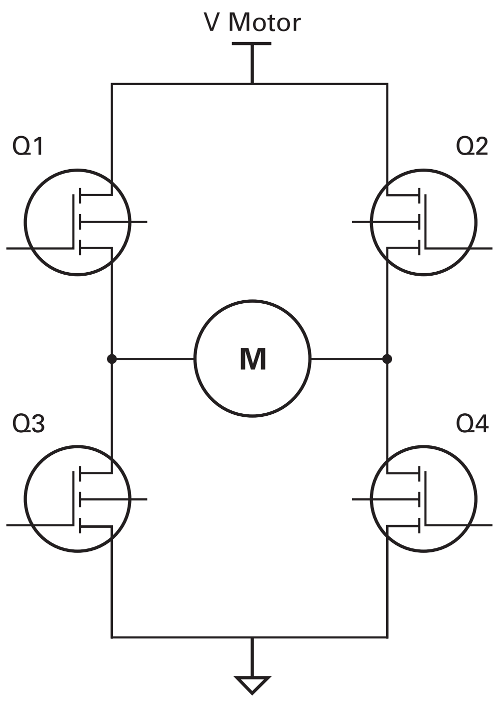

Figure 8. H-Bridge Equivalent Circuit

The H-Bridge regulates motor speed by controlling the average current applied to the motor. The pulse-width modulated signal from the Z8PE003 controls the amount of time each leg of the H-Bridge is on or off. The longer the on time, the more average current is applied, and the faster the motor spins. The LMD18200 features special logic inputs that are TTL- and CMOS-compatible to turn on the MOSFETs.

The voltage developed across R11 sets the current limit flag. The following equation determines the R11 value. Note that the value used in the calculation is the load current for this application, not the stall current, because of the near no-load condition on the motor.

```
R = 5 Volts ÷ (377 µA ÷ A x 275 mA) = 48KΩ
```

The function of the thermal overload flag is to warn the Z8PE003 when the temperature inside the H-Bridge reaches the 145∞C warning limit. Should this limit be reached, the Z8PE003 can shut down the system before the H-Bridge reaches the shut-down temperature of 170∞C.Motor

The motor is a generic, permanent magnet DC motor with a 100:1 ratio output gear train. It operates in a 4.5 to 12 VDC operating range, 75-mA no-load current, and 69 rpm no-load speed.

Pulse Width Modulation

The Z8PE003 features a 16-bit PWM timer formed by combining the T0 and T1 8-bit timers to produce timer T01. The PWM drives the H-Bridge that controls the speed of the motor. As shown in Figure 9, the PWM duty cycle ranges from 0 to 100%, where 0 indicates a total off and 100 indicates a total on. The higher the duty cycle the more energy the motor receives, and the faster it rotates.

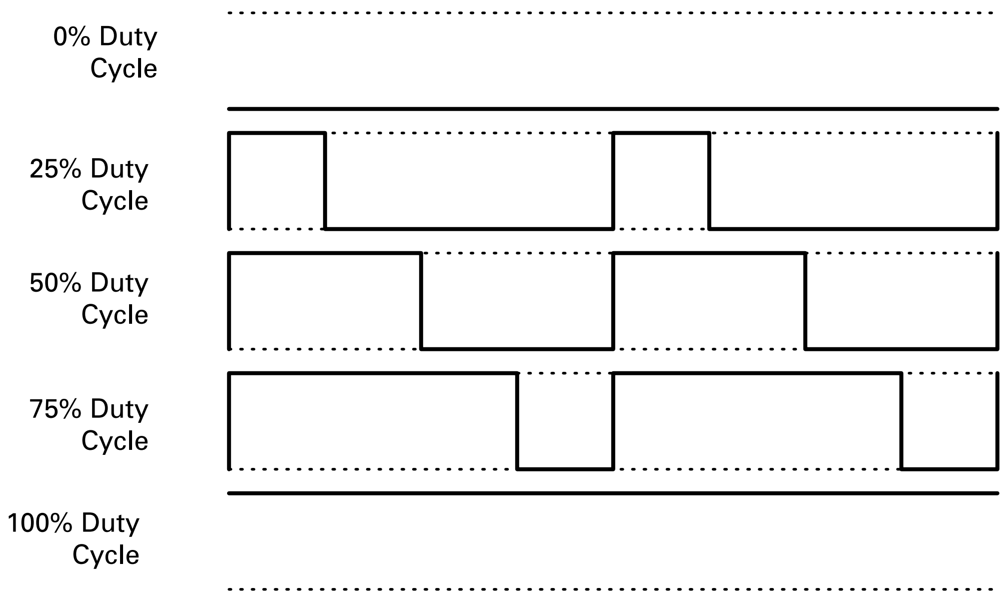

Figure 9. PWM Example

FIRMWARE IMPLEMENTATION

The Z8PE003 firmware controls the serial communication, position calculation, sensor monitor, and motor control. The code is divided into two loops: T23 and Main. Refer to the source code for a complete listing. All firmware was developed using ZiLOG Development Studio (ZDS) Version 2.0. ZDS contains a project manager, editor, assembler and linker support for the Z8PE003. ZDS also features a built-in debugger and OTP programming support.

Firmware Flags

The firmware contains several flags used throughout the code. The following sections describe each of the firmware flags.

Limit Switch Flag

The firmware uses the limit switch flag (LIM_FLAG) to determine the current state of the limit switches. Activating either limit switch sets the flag. Deactivating the limit switch clears the flag. This allows the user to stop at the limit switch and then move out of that mode with manual controls. The Main Loop section below describes limit switches.

Manual Switch Flag

The firmware uses the manual switch flag (SWITCH_FLAG) to determine the current state of the manual switches. The flag is set to `1` when the user selects OPEN-LOOP VELOCITY mode, and set to 0 when in OPEN-LOOP POSITION mode.

Current Firmware Status Flag

The current status flag (CURFLAG) incorporates many functions. The CURFLAG contains 8 bits, each of which is described below:

Bit 0. This bit is used by the firmware to determine if the user released a switch after a limit switch is activated. If no switch is released, no motion is allowed. When this occurs, the user can engage a limit switch, release the manual input switch, or restart the movement of the belt by pushing any manual input switch. The Open-Loop Manual Mode section contains more details.

Bit 1. This bit is used to communicate to the GUI that a STOP command was issued. When set to `1`, this bit indicates that a sensor issued a STOP command.

Bit 2. When set to `1`, this bit indicates a thermal limit warning.

Bit 3. When set to `1`, this bit indicates that limit switch 1 is activated.

Bit 4. When set to `1`, this bit indicates that limit switch 2 is activated.

Bit 5. When set to `1`, this bit indicates that manual switch 1 is activated.

Bit 6. When set to `1`, this bit indicates that limit switch 2 is activated.

Bit 7. When set to `1`, this bit indicates to the firmware that it is time to get a new differential constant value. When set to `0`, the PID filter continues to apply the same D constant.

Input Command Flag

The firmware uses the input command flag register (FREG) to indicate commands issued from the GUI and the current operating function of the firmware. The FREG contains 8 bits, each of which is described below:

Bit 0. This bit is set to `1` to indicate CLOSED-LOOP mode or set to `0` to indicate OPEN-LOOP MANUAL mode.

Bit 1. This bit is set to `1` to indicate that the firmware is ready to transmit a packet to the GUI.

Bit 2. This bit is set to `1` to indicate that the firmware is ready to receive a packet from the GUI.

Bit 3. This bit is set to `1` to indicate that data is available from the GUI to decode.

Bit 4. Not used.

Bit 5. This bit is set to `1` to indicate that limit switches are turned on. `Zero` indicates the switches are turned off.

Bit 6. This bit is set to `1` to indicate that manual input switches are turned on. `Zero` indicates the switches are turned off.

Bit 7. This bit is set to `1` to indicate that the STOP flag is on. Turn off the motor.

Main Loop

The main loop controls the monitoring of the thermal limit flag, manual input switches, limit switches, decoding commands from the GUI, and RS-232 transmission to the GUI. Appendix B contains a detailed flow chart of the main loop

Thermal Limit

Port pin PA6 is pulled to Low if the H-Bridge reaches its temperature warning limit. Should that occur, the firmware issues a STOP command and sets the appropriate flag to send to the GUI.

Manual Input Switches

The manual switches can be operated in two different modes: OPEN-LOOP MANUAL POSITION mode or OPEN-LOOP MANUAL VELOCITY mode.

Open-Loop Manual Position Mode

If not in CLOSED-LOOP mode, a manual input command flag (bit 5 or 6 of CURFLAG) is set when the user pushes S1 or S2. CURFLAG indicates to the GUI that the manual input switch was pushed. The firmware starts the motor motion either in CW or CCW motion at the maximum velocity (PWM equal to 100%) by setting the direction bit (PA3) and turning off the STOP flag. The firmware continually checks to see if the button is released and if so, sets the STOP flag. The motor motion stops when the button is released or a limit switch is engaged, if they are enabled.

Open-Loop Manual Velocity Mode

MANUAL VELOCITY mode is triggered when buttons S1 and S2 are pushed at the same time. When pushed and released, the motor spins at a constant velocity in a counter clockwise direction. Velocity is decreased by pushing S1 or increased by pushing S2. Pushing both buttons at the same time releases the motor from MANUAL VELOCITY mode and returns it to OPEN-LOOP MANUAL POSITION. mode. Limit switches are turned off during this mode.

Limit Switches

The limit switches are activated when the LED turns on the phototransistor. When the phototransistor is turned on, the port pin is pulled Low. The main loop verifies that the port is pulled Low and checks to see that the limit switch flag is set. If set, the main loop jumps to the CHECK STOP FLAG routine indicating that this limit switch interrupt was serviced. This allows the user to move off of a limit switch, turning the limit switch back off. If the flag is not set, then the STOP flag is issued, the motor is turned off, and the GUI command flag is set.

Decoding Commands from the GUI

The Z8PE003 receives commands from the GUI through the serial port. These commands are decoded in the main loop of the firmware. Table 3 lists a number of commands that the Z8PE003 can receive. The 8-bit command is sent in 2 packets followed by a 16-bit data value divided into 4 packets (see Table 4). The first 2 packets are the 8-bit command divided into two 4-bit packets, most significant nibble (MSN) and least significant nibble (LSN). The numerical value must be changed to an ASCII character. The same procedure is repeated for the second packet and the data packets. The data packets from the GUI can arrive in the form of a position command, proportional constant, integral constant, or differential constant.

The actual receive bit timing over the serial port is described in the T23 Loop section of this document.

Table 3. GUI to Z8PE003 Command Structure

Command Description	Command Byte	Data Value Bit Length
Load Position Command and go to position	0XXX 0001	16
Set Proportional Value	0XXX 0010	16
Set Integral Value	0XXX 0100	16
Set Differential Value	0XXX 1000	16
Limit Switches Off	0XX0 XXXX	16
Limit Switches On	0XX1 XXXX	16
Manual Input Off	0X0X XXXX	16
Manual Input On	0X1X XXXX	16
Free run	00XX XXXX	16
STOP	01XX XXXX	16

Table 4. Transmission Sequence

Command MSN	Start	D0	D1	D2	D3	D4	D5	D6	D7	End (11/2 bits)
Command LSN	Start	D0	D1	D2	D3	D4	D5	D6	D7	End (11/2 bits)
Data MSB–MSN	Start	D0	D1	D2	D3	D4	D5	D6	D7	End (11/2 bits)
Data MSB–LSN	Start	D0	D1	D2	D3	D4	D5	D6	D7	End (11/2 bits)
Data LSB–MSN	Start	D0	D1	D2	D3	D4	D5	D6	D7	End (11/2 bits)
Data LSB–LSN	Start	D0	D1	D2	D3	D4	D5	D6	D7	End (11/2 bits)

Transmitting Status to the GUI

The main loop also handles the transmission of data to the GUI. Table 5 illustrates the status and position data structure used to respond to the GUI. Packet 1 contains the command while packets 2 and 3 contain the current position value. See Appendix C for the transmit flow chart.

Table 6 illustrates the command structure. It communicates necessary sensor conditions from the Z8PE003 to the GUI. Each update is an 8-bit packet that is decoded by the GUI, followed by 16 bits of the current position value.

Table 5. Z8PE003 to GUI Sensor Data Output Structure

Sensor Description	Sensor Binary Value	Current Position Data
Current Limit Warning	0XXX XX01	16
Thermal Warning	0XXX X10X	16
Limit Switch 1 – Engaged	0XXX 1X0X	16
Limit Switch 2 – Engaged	0XX1 XX0X	16
Manual Switch Right	0X1X XX0X	16
Manual Switch Left	01XX XX0X	16

Table 6. Command Structure of Z8PE003 to GUI

Status MSN	Start	D0	D1	D2	D3	D4	D5	D6	D7	End (11/2 bits)
Status LSN	Start	D0	D1	D2	D3	D4	D5	D6	D7	End (11/2 bits)
Position MSB–MSN	Start	D0	D1	D2	D3	D4	D5	D6	D7	End (11/2 bits)
Position MSB–LSN	Start	D0	D1	D2	D3	D4	D5	D6	D7	End (11/2 bits)
Position LSB–MSN	Start	D0	D1	D2	D3	D4	D5	D6	D7	End (11/2 bits)
Position LSB–LSN	Start	D0	D1	D2	D3	D4	D5	D6	D7	End (11/2 bits)

Command Transfer Packet Timing

The Z8PE003 sends a data packet to the GUI every 130 msec. The GUI must respond within this time interval or the firmware sends the next updated packet.The Z8PE003 does not require connection to the GUI to operate in open-loop mode.

Position Calculation

The sampling rate for the encoder is determined as follows:

```
Encoder Change Rate = 69 Rev÷min x (128 x 4) pulses
÷Rev x 1 min÷60 sec = 588 pulses/sec
```

A minimum sample rate of 1,176 Hz (2 x 588), or one time every 849 µsec minimum, satisfies the Nyquist rate. The main loop samples the position every 150 µsec, which more than satisfies the Nyquist rate. The way the position is sampled and compared to the most recent sample, oversampling does not cause false data. Looking at the first bit of the most recent input and comparing it to the last bit of the current input determines the direction of movement. If the values are equal, the encoder moved in a clockwise direction. If they are not equal, the encoder moved in a counterclockwise direction. Table 7 shows the bit settings for counterclockwise motion and Table shows the bit settings for clockwise motion.

Table 7. Counterclockwise Motion, Bits not Equal

Last Input High Bit	Last Input Low Bit	Current Input High Bit	Current Input Low Bit
0	0	0	1
0	1	1	1
1	1	1	0
1	0	0	0

Table 8. Clockwise Motion, Bits Equal

Last Input High Bit	Last Input Low Bit	Current Input High Bit	Current Input Low Bit
0	0	1	0
0	1	0	0
1	1	0	1
1	0	1	1

See Appendix D for a complete flow chart for this routine.

Closed-Loop Mode

When a closed-loop command and the required position values are received, the Z8PE003 ignores all other inputs except a STOP command from the GUI. The PID filter is executed every 150 µsec on the position error value. The following section breaks down the PID algorithm and describes each component. See Appendix E for a flow chart of the PID filter.

Command Value

The GUI provides the command value, which is a 16-bit unsigned number ranging from 5 to 65535. An invalid command guard band was placed from 0 to 5 and 65530 to 65535 because of belt slack. This guard band ensures that no roll-over takes place from a command. For example, a command of 0 might overshoot if the gain is too high and the position value would be 65535. The overshoot causes the Z86E001 to start from 65535 down to 0 instead of working back up to 0.

Error Value

The firmware checks the required command and subtracts it from the actual position. This number is stored as a 16-bit unsigned number. If the error is less than 0 (command is less than current position), the direction bit is toggled High to turn the motor in the correct direction. The two's-complement is taken of the error. If the error is greater than 0, the direction bit is toggled Low. The error value is evaluated to see if it is less than five counts and, if so, the brake bit is turned on.

Proportional Constant—K_p

The proportional constant is a linear gain value, similar to an op-amp or a spring represented by Hooke's law. The position error is multiplied by the proportional constant and the result is loaded into the PWM output registers. The proportional constant is an unsigned 8-bit value ranging from 0 to 255. The maximum effect of the proportional gain is 75% of the maximum PWM value.

Integral Constant—K_i

The firmware multiplies the integral constant by the accumulated error value. The value is added to the output PWM. The maximum effect of the integral gain is 25% of the total PWM value. The integral constant acts as a wind-up function for the motor and increases its influence as time increases. The greater the error value and the longer it accumulates, the greater the velocity. For example, when an elevator starts, its velocity is slow, but as it goes higher, the velocity increases to reduce travel time.

This firmware divides the current error value by 2^9 because the accumulator saturates too quickly for demonstration purposes. The wind-up effect is not noticeable to the user. The output of the sum of error time K_i is also scaled by dividing by 2^8 to limit the output to 25% of the PWM total scale.

Differential Constant—K_d

The firmware multiplies the differential constant by the difference between the previous error and the current error. The differential influence is applied to the PID output every PID application cycle. The differential influence changes only one time every 130 msec. The influence of the differential constant works in a way that is similar to the way a shock absorber works on a car. The differential constant helps the settling time and the overshoot constants.

T23 Loop

The 16-bit timer T23 controls the T23 loop. When the T23 count reads 0, IRQ5 is issued within the Z8PE003. The T23 loop controls the serial communication and monitors the time until the K_d constant is changed in the PID filter.

Serial Timing

The serial port interface is set for a 9600 baud rate. The transmission packets include a start bit, 8 bits of data, no parity, and 1-1/2 stop bits. Each bit transferred requires a clock time of 104.2 msec (1/9600 baud). Therefore, one data packet (a total of 10.5 bits) requires an input time of 1.094 msec. Commands from the GUI to the firmware are six words, each eight data bits wide. The maximum amount of serial input time required is approximately 6.5 msec to receive a full command. The serial timing is divided into π-bit time, with T23 set at a 26.04 msec interrupt rate. The transmission time is equivalent, but transmission is done in the main loop, not T23.

SOFTWARE IMPLEMENTATION

The GUI software was developed using Microsoft Visual Basic Version 5.0. All the controls are Visual Basic standard controls, including MSCOMM, which is the serial interface control to the Z8PE003. The MSCOMM control handles all serial port functions.

The GUI allows the user to disable the limit switches and the manual input switches; to change the PID filter constants; to monitor the current sensors, position, and velocity; and to issue a POSITION command.

SUMMARY

The Z8PE003 is a powerful processor designed for complex applications. It offers excellent computation speed required for real-time, closed-loop systems.

APPENDIX A: HARDWARE SCHEMATIC

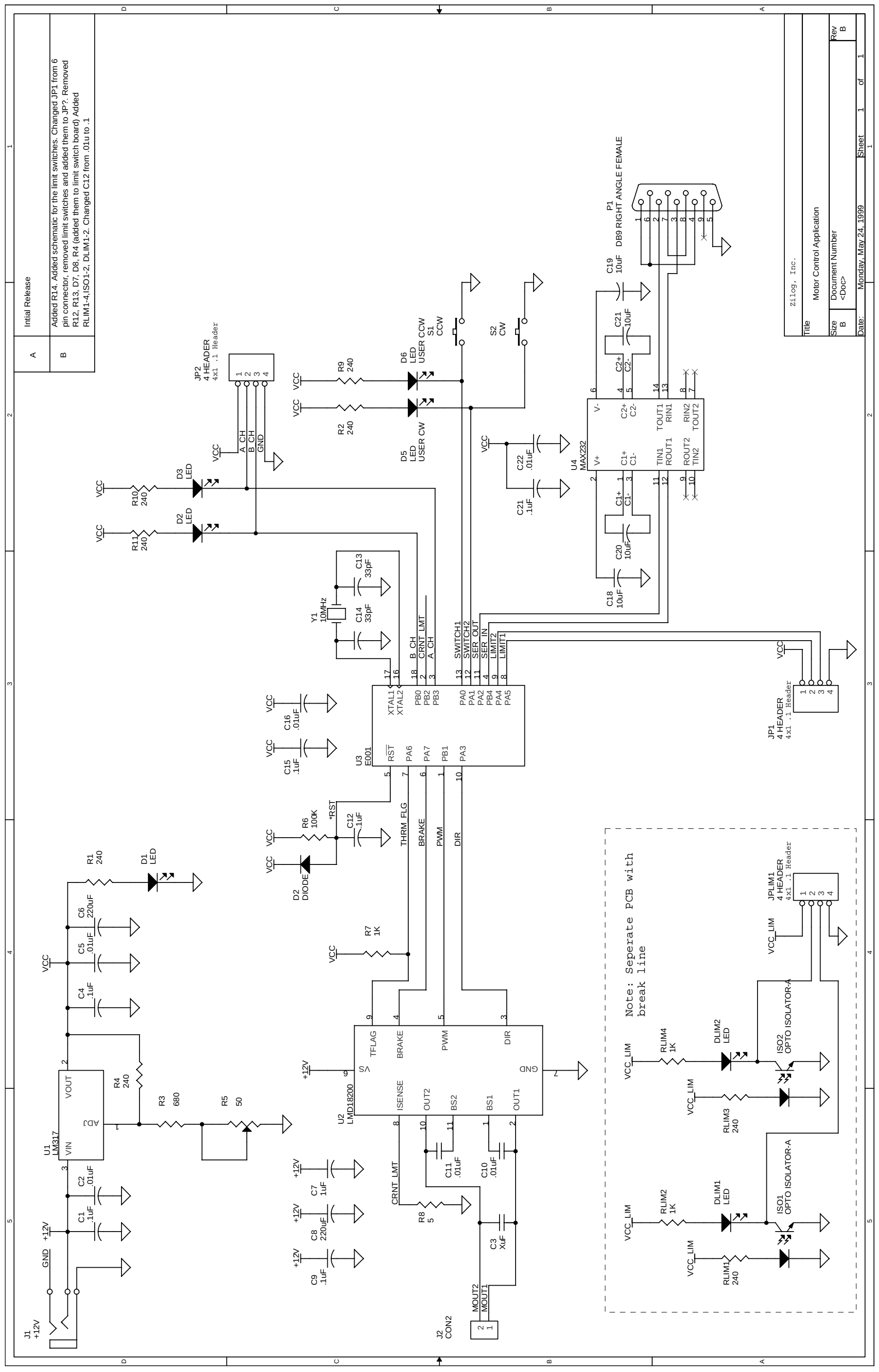
Intial Release
Added R14. Added schematic for the limit switches. Changed JP1 from 6 pin connector, removed limit switches and added them to JP?. Removed R12, R13, D7, D8, R4 (added them to limit switch board) Added RLIM1-4,ISO1-2, DLIM1-2. Changed C12 from .01u to .1
Zilog, Inc.
Motor Control Application
Monday, May 24, 1999
P1 DB9 RIGHT ANGLE FEMALE
U4 MAX232
U3 E001
U2 LMD18200
U1 LM317
JP1 4 HEADER
JP2 4 HEADER
JPLIM1 4 HEADER
Y1 10MHz
ISO1 OPTO ISOLATOR-A
ISO2 OPTO ISOLATOR-A
Note: Seperate PCB with break line
J1 +12V
J2 CON2

Figure 10. Hardware Schematic

APPENDIX B: MAIN LOOP FLOW CHART

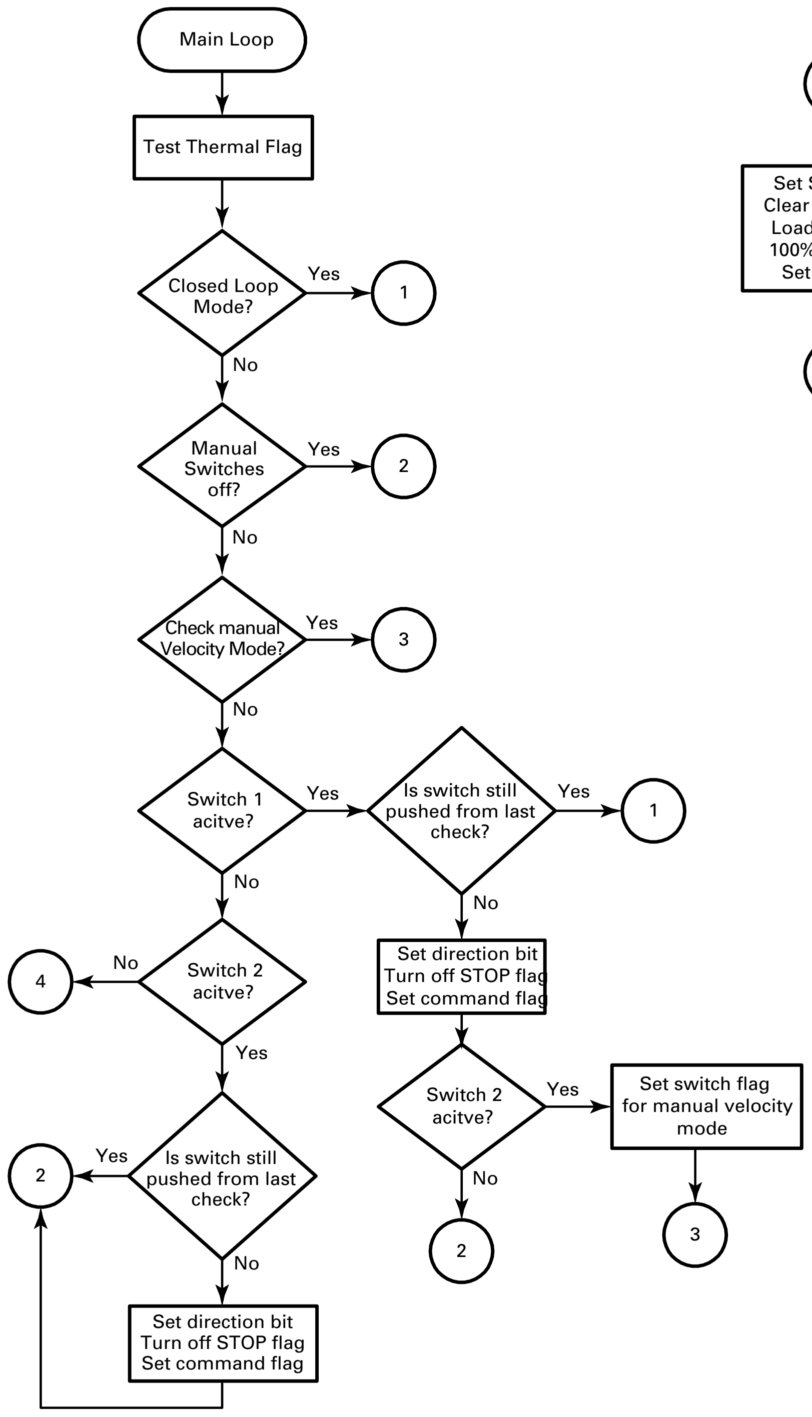

Figure 11. Main Loop Flow Chart, Part 1 of 4

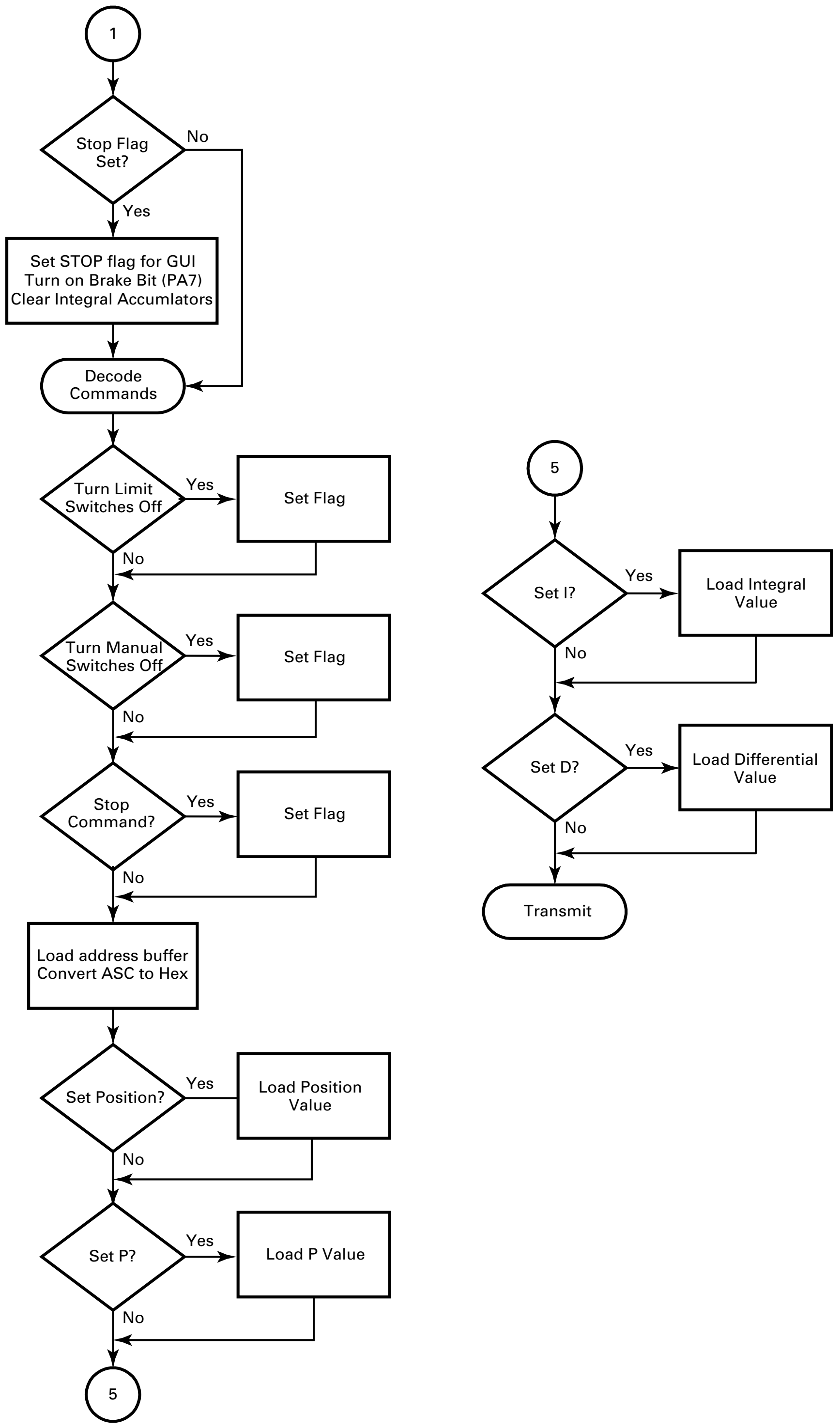

Figure 12. Main Loop Flow Chart, Part 2 of 4

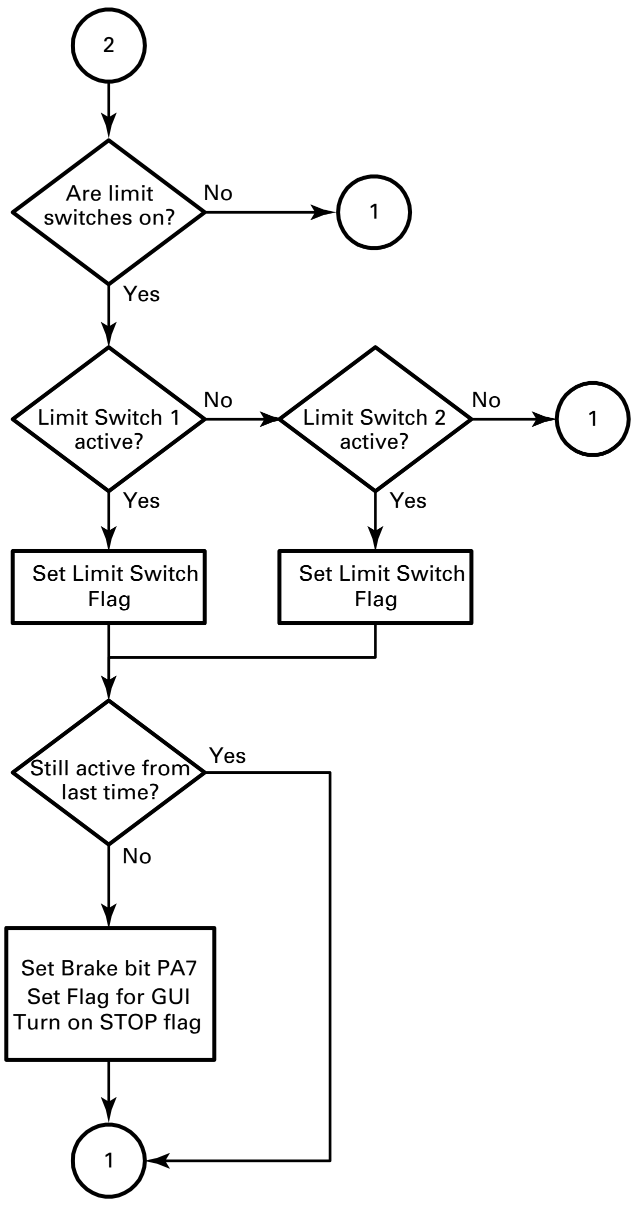

Figure 13. Main Loop Flow Chart, Part 3 of 4

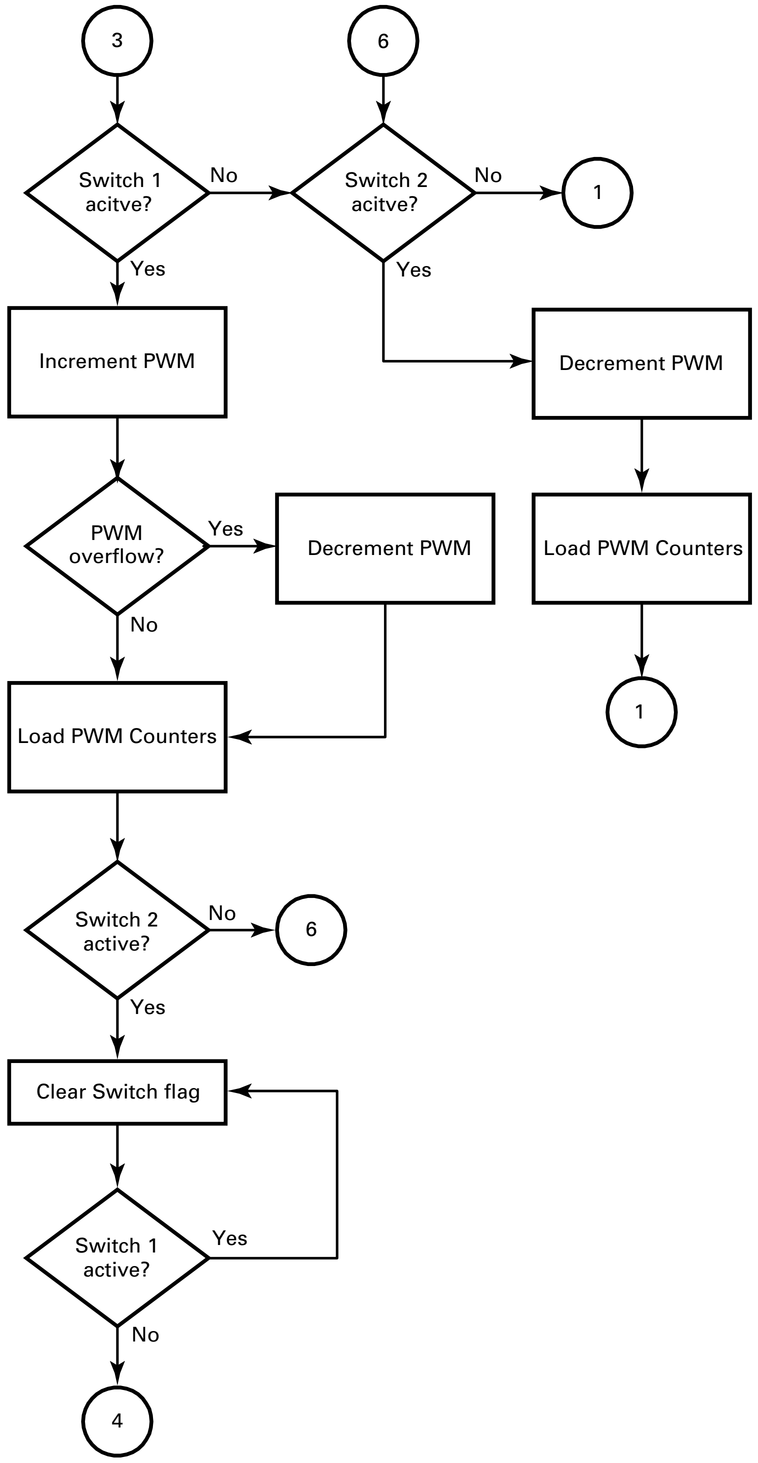

Figure 14. Main Loop Flow Chart, Part 4 of 4

APPENDIX C: TRANSMIT FLOW CHART

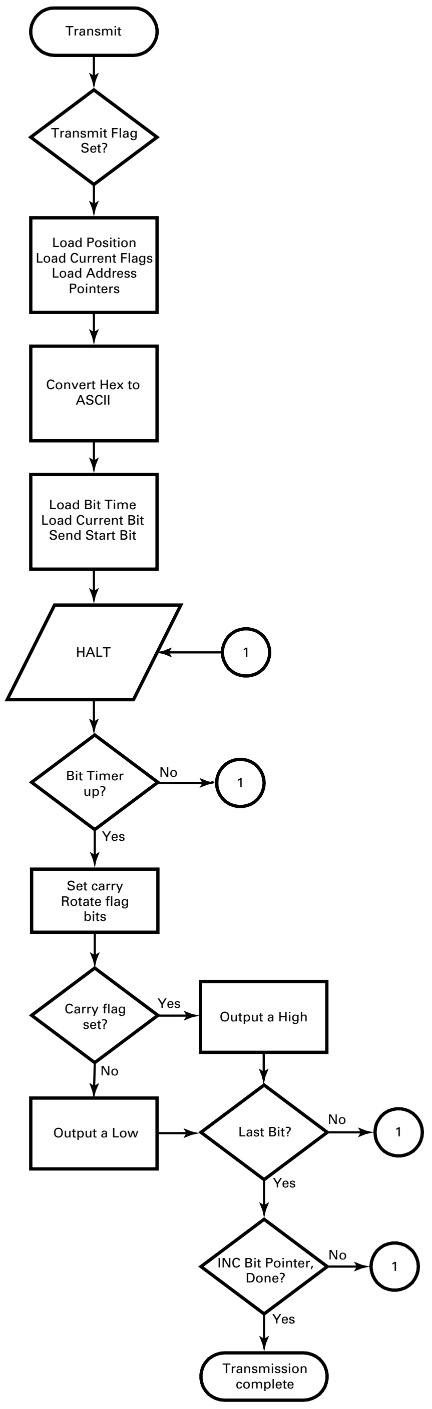

Figure 15. Transmit Flow Chart

APPENDIX D: POSITION CALCULATION FLOW CHART

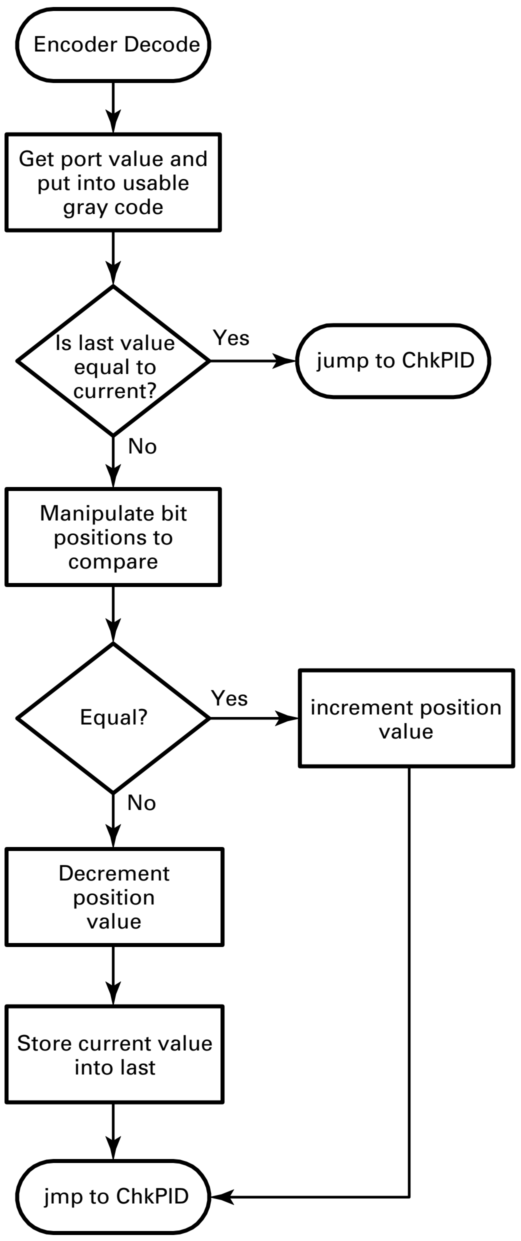
Encoder Decode
Get port value and put into usable gray code
Is last value equal to current?
Yes
jump to ChkPID
No
Manipulate bit positions to compare
Equal?
Yes
increment position value
No
Decrement position value
Store current value into last
jmp to ChkPID

Figure 16. Position Calculation Flow Chart

APPENDIX E: PID LOOP FLOW CHART

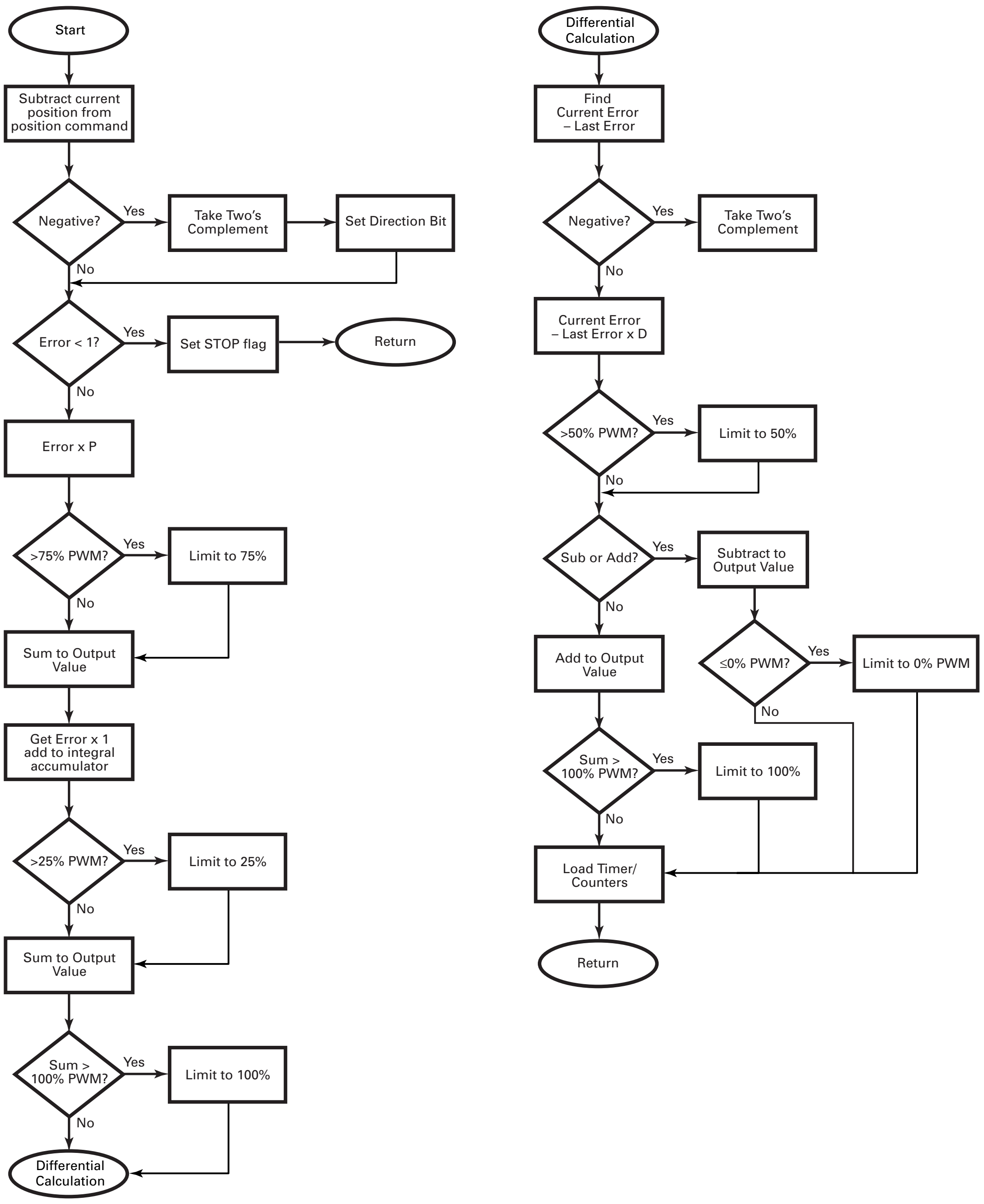

Figure 17. PID Loop Flow Chart

INFORMATION INTEGRITY

DOCUMENT DISCLAIMER